Aplikacje mikrokontrolerów

Projekt

# Platforma multimedialna wykorzystująca komunikację PC – mikrokontroler za pomocą przejściówki USB / RS-232 (implementacja USB – CDC)

Autor: **Wojciech Gałecki**

Rok akademicki 2019/2020

Prowadzący:

**dr inż. Łukasz Krzak**

**mgr inż. Jan Macheta**

# Krótki opis projektu

Tematem projektu jest platforma złożona z trzech układów elektronicznych wykorzystujących komunikację STM32 <-> Komputer PC. Zostały one zmontowane przy użyciu płytek stykowych i podłączone do płytki STM32F411E Discovery Board, której serce stanowi mikrokontroler STM32F411VET6. Układy są sterowane za pomocą poleceń wysyłanych z terminala obsługującego połączenie z portem szeregowym. Taka komunikacja może być realizowana za pomocą kabla USB przy użyciu urządzenia klasy USB - CDC, które posiada funkcję tworzenia wirtualnego portu COM w celu imitacji protokołu RS-232. W tym wypadku takie urządzenie stanowi płytka z mikrokontrolerem.

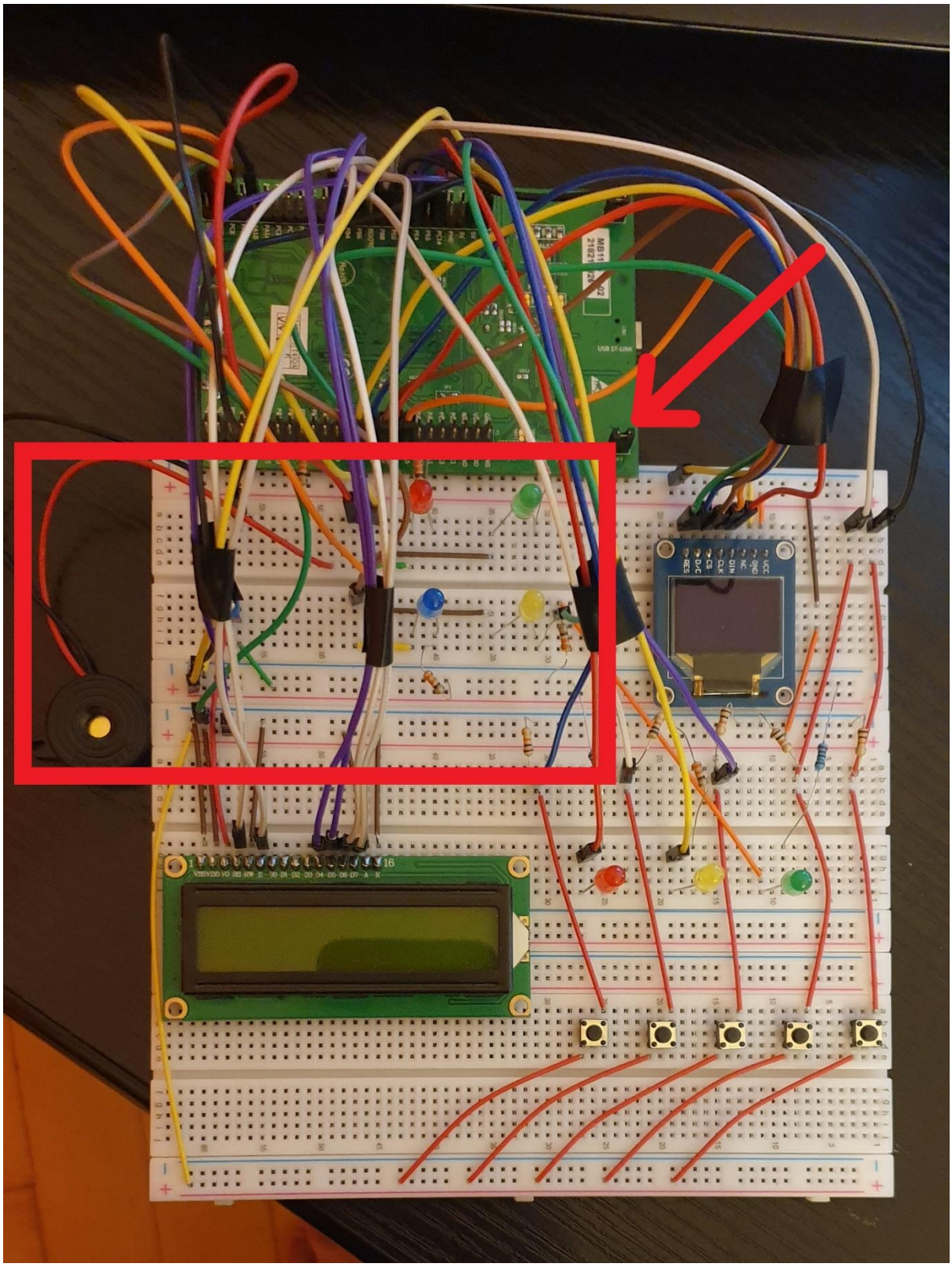
# Użyte układy i elementy elektroniczne

- Płytka STM32F411E Discovery Board oparta na mikrokontrolerze STM32F411VET6
- Kolorowy wyświetlacz OLED 96x64 px oparty na sterowniku SSD1331 i komunikacji przez SPI
- Wyświetlacz LCD 2x16 znaków oparty na sterowniku HD44780
- Buzzer z generatorem
- 5 przycisków typu tact-switch
- Rezystory: 6 x 330 Ω, 5 x 10 kΩ, 2 x 1.5 kΩ
- Diody LED 5mm : 2 x czerwona, 2 x żółta, 2 x zielona, 1 x niebieska
- 3 płytki stykowe (830 otworów)
- Przewody: zasilający (A <-> Mini B), USB (A <-> Micro B), połączeniowe żeńsko-męskie

Schemat elektryczny został załączony do niniejszej dokumentacji w formatach .png i .svg .

# Opis układów stanowiących temat projektu

**Układ testowy (4 diody LED + buzzer)**



Układ składa się z 4 diod LED oraz buzzera z generatorem. Jest to najprostszy z dostępnych układów i umożliwia zapalenie / zgaszenie diod LED oraz włączenie / wyłączenie buzzera.

Komendy dostępne wyłącznie przy aktywowanym układzie testowym:

"**RED LED ON**" / "**RED LED OFF**" – zapalenie / zgaszenie czerwonej diody

"**GREEN LED ON**" / "**GREEN LED OFF**" – zapalenie / zgaszenie zielonej diody

"**BLUE LED ON**" / "**BLUE LED OFF**" - zapalenie / zgaszenie niebieskiej diody

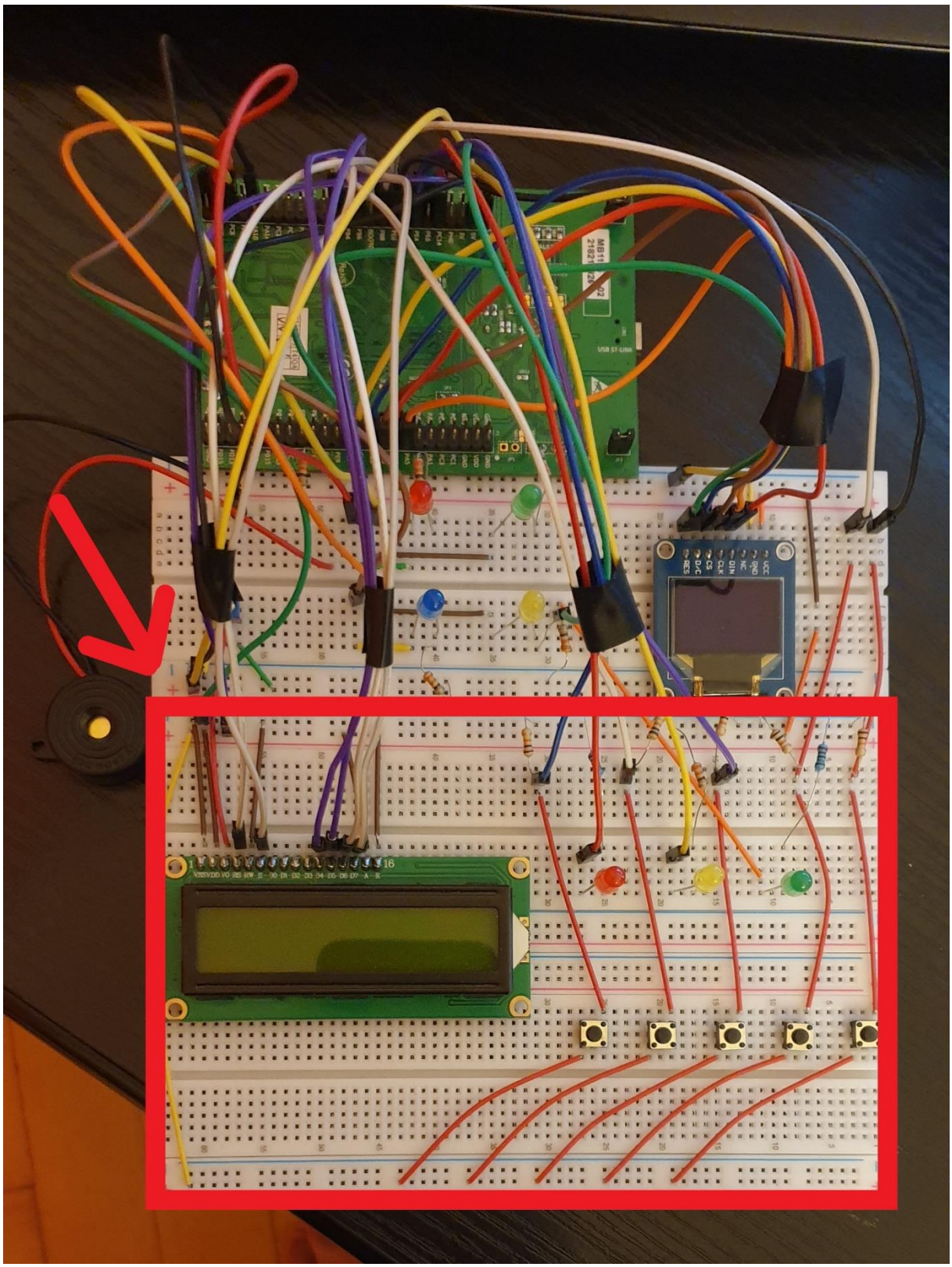
"**YELLOW LED ON**" / "**YELLOW LED OFF**" – zapalenie / zgaszenie żółtej diody

"**ALARM ON**" / "**ALARM OFF**" - uruchomienie / wyłączenie buzzera

Pliki źródłowe powiązane z układem:

- ***Core/Src/TestCircuit.c***
- ***Core/Inc/TestCircuit.h***

**Układ z wyświetlaczem LCD 2x16 i przyciskami typu tact-switch (+ 3 diody LED)**



Układ składa się z wyświetlacza LCD 2x16 znaków, opartego na sterowniku HD44780, oraz pięciu przycisków typu tact-switch i trzech diod LED. Umożliwia on realizację dwustronnego wysyłania wiadomości pomiędzy komputerem PC a mikrokontrolerem. W przypadku wysłania wiadomości z komputera do STM32, układ zapamiętuje je (do 10 wiadomości) i umożliwia ich późniejszy odczyt przez użytkownika. Z kolei przesyłanie wiadomości do komputera jest realizowane za pomocą przycisków, dzięki którym użytkownik może stworzyć swoją wiadomość i ją wysłać. Każdy z przycisków posiada dwie funkcje, jedną realizowaną za pomocą krótkiego wciśnięcia (do 2 sekund), a drugą przy pomocy dłuższego naciśnięcia.

Dostępne przyciski (widoczne od lewej na zdjęciu):

**READ / WRITE**

**Krótkie wciśnięcie:** odczyt kolejnej nieprzeczytanej wiadomości; w przypadku braku takich wiadomości wyświetlenie stosownego komunikatu na wyświetlaczu

**Długie wciśnięcie:** przejście do trybu nadawania (zapalenie żółtej diody LED, wyczyszczenie ekranu i powrót kursora do początkowej pozycji)

**BACKSPACE / CLEAR**

**Krótkie wciśnięcie:** usunięcie ostatniego znaku (aktualnie widocznego lub poprzedniego w przypadku braku aktualnie widocznego znaku)

**Długie wciśnięcie:** usunięcie całej wiadomości i powrót kursora do początkowej pozycji

**LEFT / CAPS-LOCK**

**Krótkie wciśnięcie:** zmiana aktualnie wyświetlanego znaku na poprzedni w swojej grupie (małe litery, wielkie litery, cyfry)

**Długie wciśnięcie:** zmiana aktualnej grupy znaków z małych liter na wielkie i vice versa; w przypadku, gdy aktualną grupą znaków są cyfry, żadna akcja nie jest podejmowana

**RIGHT / ALPHA <-> NUM SWITCH**

**Krótkie wciśnięcie:** zmiana aktualnie wyświetlanego znaku na poprzedni w swojej grupie (małe litery, wielkie litery, cyfry)

**Długie wciśnięcie:** zmiana aktualnej grupy znaków z małych lub wielkich liter na cyfry lub z cyfr na małe litery

**OK / SEND**

**Krótkie wciśnięcie:** zatwierdzenie aktualnie wyświetlanego znaku i przejście kursora do następnej pozycji

**Długie wciśnięcie:** wysłanie wiadomości do komputera PC, wyczyszczenie ekranu i powrót kursora do początkowej pozycji

Diody LED wchodzące w skład układu:

- **czerwona** → zapalona, jeśli pozostały wiadomości nieodczytane przez użytkownika; w przeciwnym razie zgaszona
- **żółta** → zapalona, jeśli układ znajduje się obecnie w trybie nadawania; w przeciwnym razie zgaszona
- **zielona** → zapalona przez 2 sekundy po każdym wysłaniu wiadomości, za jej zgaszenie po określonym czasie odpowiada timer ustawiony w trybie OPM (One Pulse Mode)

Komendy dostępne wyłącznie przy aktywowanym układzie z wyświetlaczem LCD:

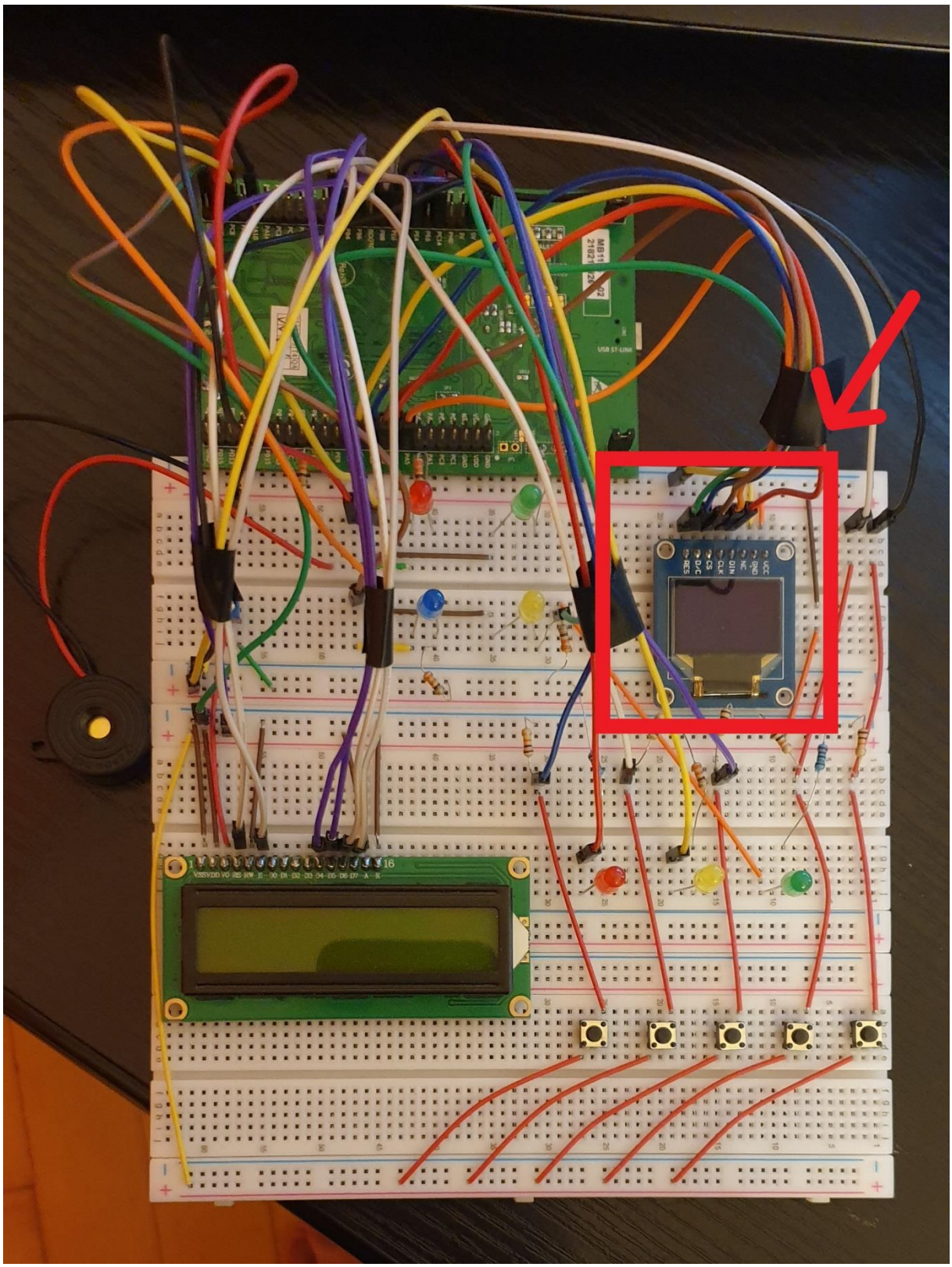
"**SEND '[TEKST]'**" – wysyła wiadomość do mikrokontrolera, która może być następnie odczytana przy pomocy krótkiego wciśnięcia przycisku READ / WRITE. Wiadomość nie może przekraczać 32 znaków oraz zawierać znaku apostrofu.

"**GET UNREAD MESSAGES NUMBER**" – wyświetla w terminalu ilość pozostałych nieprzeczytanych wiadomości.

Pliki źródłowe powiązane z układem:

- ***Core/Src/LCDCircuit.c***
- ***Core/Inc/LCDCircuit.h***
- ***Core/Src/LiquidCrystal.c***
- ***Core/Inc/LiquidCrystal.h***

**Układ z wyświetlaczem OLED (gra Snake – sterowanie za pomocą klawiatury PC)**



Układ składa się z kolorowego wyświetlacza OLED 96x64 px opartego na sterowniku SSD1331 i wykorzystującego komunikację przez interfejs SPI. Za jego pomocą wyświetlana jest implementacja gry Snake. Użytkownik ma do dyspozycji 3 poziomy trudności oraz może przerwać rozgrywkę w dowolnym momencie za pomocą komendy. Układ może znajdować się w trzech stanach – ekran startowy, rozgrywka oraz ekran końcowy. Sterowanie wężem jest możliwe za pomocą przycisków WSAD klawiatury komputera PC - w tym celu należy korzystać z terminala ustawionego w tryb wysyłania pojedynczych znaków bez potrzeby ich zatwierdzenia, co znacznie utrudniłoby rozgrywkę.

Komendy dostępne wyłącznie przy aktywowanym układzie z wyświetlaczem OLED:

Dostępne podczas wyświetlania ekranu startowego lub końcowego:

"**START**" – rozpoczęcie rozgrywki

"**SET DIFFICULTY EASY**" / "**SET DIFFICULTY MEDIUM**" / "**SET DIFFICULTY HARD**" – zmiana poziomu trudności nadchodzącej rozgrywki (odstęp czasowy pomiędzy kolejnymi ruchami węża: poziom łatwy – 200 ms, poziom średni – 100 ms, poziom trudny – 50 ms)

"**GET DIFFICULTY**" – wyświetlenie poziomu trudności nadchodzącej rozgrywki

Dostępne w czasie rozgrywki:

"**w**" – zmiana kierunku poruszania się węża - w górę

"**s**" – zmiana kierunku poruszania się węża - w dół

"**a**" – zmiana kierunku poruszania się węża - w lewo

"**d**" – zmiana kierunku poruszania się węża - w prawo

"**STOP**" – przerwanie rozgrywki i powrót do ekranu startowego

Pliki źródłowe powiązane z układem:

- ***Core/Src/OLEDCircuit.c***
- ***Core/Inc/OLEDCircuit.h***
- ***Core/Src/SSD1331.c***
- ***Core/Inc/SSD1331.h***
- ***Core/Src/Fonts.c***
- ***Core/Inc/Fonts.h***
- ***Core/Inc/MacroAndConst.h***
- ***Core/Inc/LIB_Config.h***

**Komendy dostępne przy dowolnym aktywowanym układzie**

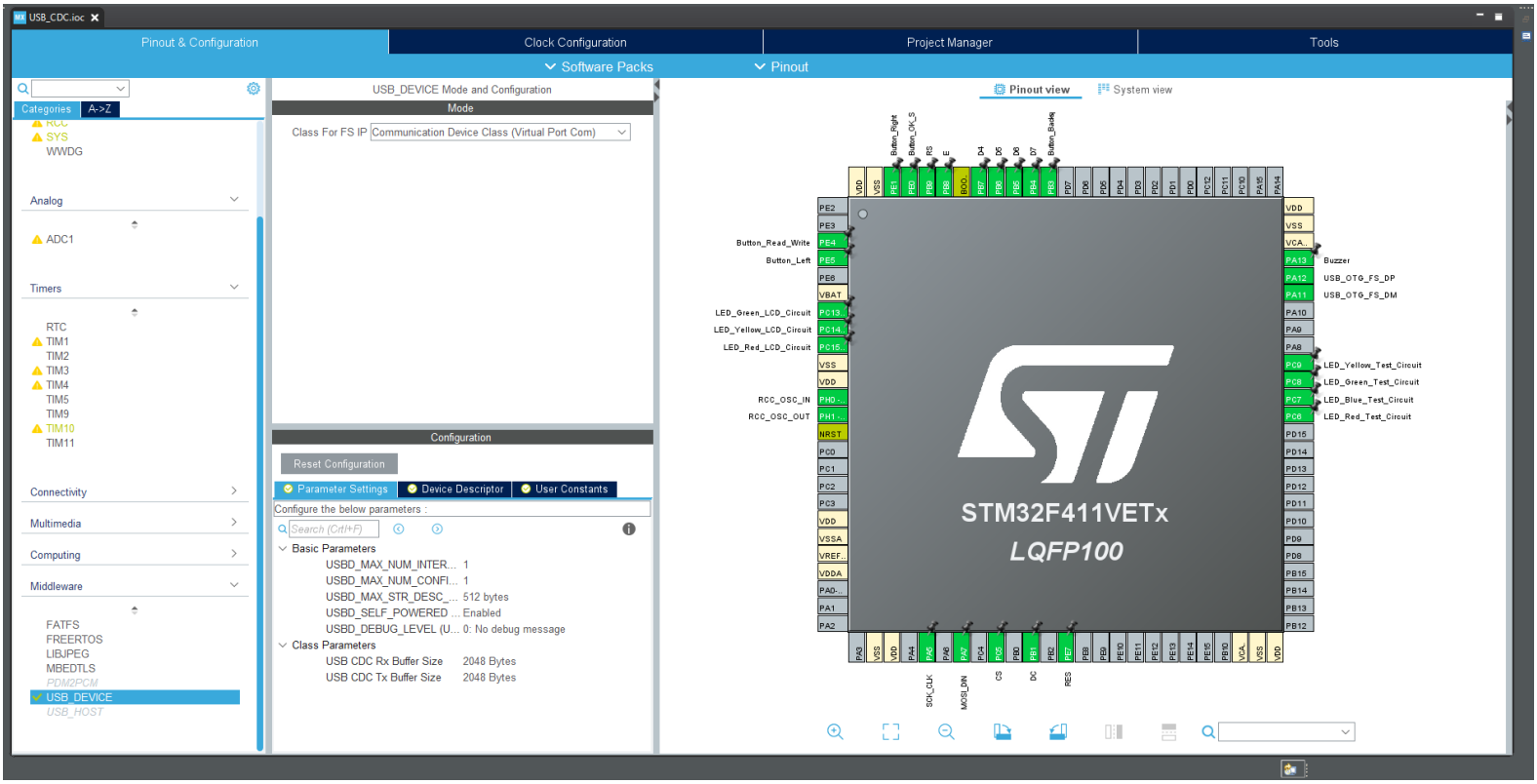
“**SWITCH CIRCUIT TEST**” / „**SWITCH CIRCUIT LCD**” / “**SWITCH CIRCUIT OLED**” – zmiana aktywnego układu na wyspecyfikowany w komendzie i wyłączenie pozostałych układów

“**GET COMMANDS**” – wyświetlenie w terminalu komend dostępnych dla obecnie aktywowanego układu

“**GET ALL COMMANDS**” – wyświetlenie w terminalu wszystkich komend dostępnych w projekcie

# Opis kodu źródłowego

Projekt został wykonany przy użyciu pakietu STM32CubeIDE, które łączy zintegrowane środowisko programistyczne TrueStudio z narzędziem STM32CubeMX, umożliwiającym automatyczną generację kodu po uprzedniej konfiguracji projektu przez użytkownika.




Widok STM32CubeMX w STM32CubeIDE

## Konfiguracja projektu w narzędziu STM32CubeMX

Piny **PA12** i **PA11** odpowiadają za komunikację w standardzie USB On The GO, zaś piny **PA7** i **PA5** za obsługę interfejsu SPI. Piny **PH0 – OSC - IN** i **PH1 – OSC- OUT** są połączone z 8 MHz rezonatorem kwarcowym znajdującym się na płytce Discovery, który został wykorzystany do taktowania zegarami w projekcie. Piny **PE5**, **PE4**, **PE1**, **PE0** i **PB3** zostały podłączone do przycisków typu tact-switch i są ustawione w tryb GPIO_EXTI umożliwiający obsługę przerwań zewnętrznych. Pozostałe piny używane w układzie zostały ustawione w tryb GPIO_OUTPUT. Wszelkie podciągnięcia do zasilania lub masy zostały zrealizowane w sposób sprzętowy na płytkach stykowych.

Szczegółowe ustawienia dotyczące projektu mogą zostać odczytane przy uruchomieniu pliku USB_CDC.ioc za pomocą narzędzia STM32CubeMX. Są one oczywiście również widoczne w samym kodzie źródłowym.

## Podział kodu pod względem autorstwa

Cały kod źródłowy projektu można podzielić na 3 grupy:

- kod stworzony przez autora projektu
- kod pochodzący z zewnętrznych bibliotek (pobranych z Internetu)
- kod wygenerowany przez STM32CubeMX

*Kod stworzony przez autora projektu:*

- ***Core/Src/TestCircuit.c*** **|** ***Core/Inc/TestCircuit.h*** → całość plików
- ***Core/Src/LCDCircuit.c*** **|** ***Core/Inc/LCDCircuit.h*** → całość plików
- ***Core/Src/OLEDCircuit.c*** **|** ***Core/Inc/OLEDCircuit.h*** → całość plików
- ***Core/Src/main.c*** **|** ***Core/Inc/main.h*** → kod otoczony znacznikami **/* USER CODE [...] */**
- ***USB_DEVICE/App/usbd_cdc_if.c*** **|** ***USB_DEVICE/App/usbd_cdc_if.h*** → kod otoczony znacznikami **/* USER CODE [...] */**

Kod pochodzący z zewnętrznych bibliotek (pobranych z Internetu):

Biblioteka do obsługi wyświetlacza LCD 2x16 (autor: S. Saeed Hosseini, źródło: https://github.com/SayidHosseini/STM32LiquidCrystal):

- ***Core/Src/LiquidCrystal.c*** **|** ***Core/Inc/LiquidCrystal.h***

Biblioteka do obsługi wyświetlacza OLED (autor: firma Waveshare - producent wyświetlacza, źródło: https://www.waveshare.com/wiki/File:0.95inch-RGB-OLED-Code.zip):

- ***Core/Src/SSD1331.c* | *Core/Inc/SSD1331.h***
- ***Core/Src/Fonts.c* | *Core/Inc/Fonts.h***
- ***Core/Inc/LIB_Config.h***
- ***Core/Inc/MacroAndConst.h***

Pozostała część kodu źródłowego została wygenerowana przez narzędzie STM32CubeMX po uprzedniej konfiguracji przez autora projektu.

## Krótki opis autorskiego kodu

***Core/Src/TestCircuit.c***

*void TestCircuitCommandHandler(void)* → wykonanie określonej akcji w układzie testowym, zależnej od komendy wydanej przez użytkownika

***Core/Src/LCDCircuit.c***

*void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)* → przerwanie timera działającego w trybie OPM (One Pulse Mode) odpowiadające za zgaszenie zielonej diody 2 sekundy po wysłaniu wiadomości przez użytkownika

*void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)* → przerwanie pochodzące od pinów podłączonych do przycisków typu tact-switch, które odpowiada za wykrycie naciśnięcia i zwolnienia przycisku (zaimplementowana została również logika odpowiadająca za debouncing)

*void ButtonTrigger(void)* → wykonanie odpowiedniej akcji realizowanej po naciśnięciu jednego z przycisków

*void LCDCircuitCommandHandler(void)* → wykonanie określonej akcji w układzie z wyświetlaczem LCD, zależnej od komendy wydanej przez użytkownika

***Core/Src/OLEDCircuit.c***

*void StartGame(SnakeBodyElement **headRef)* → rozpoczęcie rozgrywki - konfiguracja gry, stworzenie listy jednokierunkowej, wyświetlenie obiektów na wyświetlaczu

*void AddSnakeBodyElement(SnakeBodyElement *head)* → dodanie kolejnego elementu ciała węża

*void MoveForwards(SnakeBodyElement *head, Direction dir)* → ruch węża o jedną pozycję do przodu w aktualnie ustawionym kierunku poruszania się

*void CreateNewApple(SnakeBodyElement *head)* → stworzenie nowego jabłka na mapie gry

*void GameOver(SnakeBodyElement **headRef, GameState nextGameState)* → zakończenie rozgrywki – wyczyszczenie ekranu, zwolnienie pamięci, wyświetlenie ekranu początkowego / końcowego

*void DrawWall(void)* → wyświetlenie ścian bocznych

*void DrawSnake(SnakeBodyElement *head)* → wyświetlenie węża

*void EraseSnake(SnakeBodyElement *head)* → usunięcie węża z wyświetlacza

*void OLEDCircuitAction(void)* → funkcja uruchamiana w pętli, która odpowiada za wyzwalanie ruchu węża naprzód podczas rozgrywki lub wyświetlenie zawartości ekranu początkowego / końcowego

*void OLEDCircuitCommandHandler(void)* → wykonanie określonej akcji w układzie z wyświetlaczem OLED, zależnej od komendy wydanej przez użytkownika

***Core/Src/main.c***

*void ClearBuffers(void)* → wyczyszczenie bufora nadawczego i odbiorczego

*void SwitchCircuit(void)* → zmiana aktualnie aktywnego układu na wyspecyfikowany w komendzie i wyłączenie pozostałych układów

*void GetAllCommands(void)* → wyświetlenie w terminalu wszystkich komend dostępnych w projekcie

*int main(void)* → główna funkcja programu zawierająca nieskończoną pętlę główną, uruchamiającą określone funkcje w zależności od obecnie aktywnego układu i akcji użytkownika

***USB_DEVICE/App/usbd_cdc_if.c***

*static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len)* → obsługa danych odebranych przez interfejs USB

*uint8_t CDC_Transmit_FS(uint8_t* Buf, uint16_t Len)* → przygotowanie danych do wysłania przez interfejs USB